Aufgabe 1: (18 Punkte)

Bei den Multiple-Choice-Fragen ist jeweils nur **eine** richtige Antwort eindeutig anzukreuzen. Auf die richtige Antwort gibt es die angegebene Punktzahl.

Wollen Sie eine Multiple-Choice-Antwort korrigieren, streichen Sie bitte die falsche Antwort mit drei waagrechten Strichen durch (☒)und kreuzen die richtige an.

Lesen Sie die Frage genau, bevor Sie antworten.

a) Was ist ein Stack-Frame? 2 Punkte

- ❏ Der Speicherbereich, in dem der Programmcode einer Funktion abgelegt ist.
- ❏ Ein spezieller Registersatz des Prozessors zur Bearbeitung von Funktionen.
- ❏ Ein Fehler, der bei unberechtigten Zugriffen auf den Stack-Speicher entsteht.
- ❏ Ein Bereich des Speichers, in dem u.a. lokale automatic-Variablen einer Funktion abgelegt sind.

b) Gegeben ist folgender Ausdruck: 2 Punkte

```
if ( (a = 5) || (b != 3) ) ...
```

Welche Aussage ist richtig?

- ❏ Der Wert von a hat keinen Einfluss auf das Ergebnis.
- ❏ Falls a den Wert 5 und b den Wert 7 enthält, wird falsch zurückgeliefert.
- ❏ Falls a den Wert 7 und b den Wert 5 enthält, wird falsch zurückgeliefert.
- ❏ Der Compiler meldet einen Fehler, weil dieser Ausdruck nicht zulässig ist.

c) Welche Aussage zu globalen Variablen ist richtig? 2 Punkte

- ❏ Globale Variablen sind bei sehr großen Programmen vorteilhaft, weil man alle Variablendefinitionen in einer Datei an einer Stelle hinschreiben kann und man sie dadurch sehr schnell findet.
- ❏ Durch die Verwendung von globalen Variablen kann man den Einsatz von Funktionsparametern vermeiden. Dadurch werden Programme übersichtlicher und leichter wartbar.
- ❏ Durch den Einsatz von globalen Variablen werden vor allem große Programme unübersichtlich und auf Dauer schwer wartbar, da der direkte Bezug zwischen Daten und den Funktionen verloren geht.
- ❏ Mit Hilfe von globalen Variablen kann man Nebenläufigkeitsprobleme vermeiden.

d) Welche Aussage zu Zeigern ist richtig? 2 Punkte

- ❏ Ein Zeiger darf nie auf seine eigene Speicheradresse verweisen.
- ❏ Der Speicherbedarf eines Zeigers ist unabhängig von der Größe des Objekts, auf das er zeigt.
- ❏ Ein Zeiger kann zur Manipulation von schreibgeschützten Datenbereichen verwendet werden.
- ❏ Zeiger vom Typ `(void *)` sind am besten für Zeigerarithmetik geeignet, da sie kompatibel zu jedem Zeigertyp sind.

e) Welche Aussage zu *volatile* ist richtig? 2 Punkte

- ❏ Das Schlüsselwort *volatile* unterbindet Optimierungen an einer damit deklarierten Variable.
- ❏ Das Schlüsselwort *volatile* unterbindet alle Nebenläufigkeitsprobleme.
- ❏ Das Schlüsselwort *volatile* hat nur noch eine historische Bedeutung und ist in heutigen Programmen nicht mehr nötig.
- ❏ Das Schlüsselwort *volatile* erlaubt dem Compiler bessere Optimierungen durchzuführen.

f) Wann kann es zu Nebenläufigkeitsproblemen kommen? 2 Punkte

- ❏ Wenn die Programmabschnitte im if- und else-Teil einer bedingten Anweisung auf diesselbe Variable zugreifen.
- ❏ Wenn ein Programm in einer Funktion mehrere lokale Variablen verwendet.
- ❏ Wenn aus dem Hauptprogramm und einer Unterbrechungsbehandlungsfunktion dieselbe globale Variable geschrieben wird.
- ❏ Wenn ein Programmabschnitt in einer Schleife mehrfach durchlaufen wird.

g) Wie löst man das Nebenläufigkeitsproblem "lost-update" zwischen dem Hauptprogramm und einem Interrupthandler auf einem Mikrocontroller? 2 Punkte

- ❏ Durch die Verwendung von pegelgesteuerten anstelle von flankengesteuerten Interrupts
- ❏ Durch Synchronisation mittels kurzzeitigem Sperren der Interrupts
- ❏ Die Verwendung des Schlüsselwortes volatile löst alle Nebenläufigkeitsprobleme
- ❏ Durch den Aufruf einer Callback-Funktion im Interrupthandler

h) In welchen Situationen wird ein Prozess in den Zustand "blockiert" versetzt? 2 Punkte

- ❏ während der Prozesserzeugung solange die Verwaltungsstrukturen noch nicht angelegt sind
- ❏ bei jedem Aufruf der V-Operation eines Semaphors
- ❏ wenn er ausgeführt werden könnte, aber ein anderer Prozess die CPU zugeteilt bekommen soll
- ❏ beim Lesen eines Zeichens von der Tastatur, so lange keine Taste gedrückt wird

i) Welche Angaben enthält ein Eintrag eines Katalogs (Verzeichnisses) in einem Standard-UNIX-Dateisystem? 2 Punkte

- ❏ Blocknummer des Inode-Plattenblocks und Dateiname
- ❏ Inode-Nummer und Dateiname
- ❏ Dateiname, Dateigröße, Eigentümer und Zugriffsrechte
- ❏ nur die Inode-Nummer

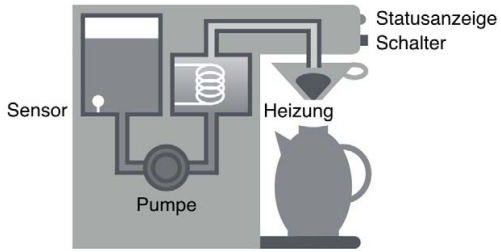
Aufgabe 2a: Kaffeemaschine (30 Punkte)

Sie dürfen diese Seite zur besseren Übersicht bei der Programmierung heraustrennen!

Schreiben Sie ein Programm für den AVR-Mikrocontroller, das eine Steuerung für eine Kaffeemaschine realisiert. Die Kaffeemaschine soll auf Tastendruck eine Kanne Kaffee produzieren und sich ausschalten, sobald kein Wasser mehr im Wassertank vorhanden ist oder der Benutzer erneut die Taste drückt.

Im Detail soll Ihr Programm wie folgt funktionieren:

- Initialisieren Sie die Hardware in der Funktion **`void init(void)`** so, dass sowohl Pumpe als auch Heizung ausgeschaltet sind. Treffen Sie keine Annahme über den initialen Zustand der Hardware-Register.
- Zu Beginn befindet sich die Kaffeemaschine im Standby-Modus. Für die Anzeige des Betriebsstatus steht die Funktion **`void setLEDState(LEDState state)`** zur Verfügung. Als Parameter für **`state`** können ihr die Betriebszustände **`Standby`**, **`Active`** und **`NoWater`** übergeben werden.
- Durch Tastendruck kann der Benutzer die Kaffeemaschine einschalten. Sie prüft dann zunächst den Füllstand des Wassertanks über den eingebauten Sensor.
- Ist Wasser vorhanden, dann aktiviert sie Pumpe und Heizung und zeigt ihre Aktivität per Statusanzeige an. Andernfalls wird zwei Sekunden lang per LED angezeigt, dass der Wassertank leer ist. Anschließend wechselt die Anzeige wieder in den Standby-Modus. Implementieren Sie hierzu eine aktive Wartefunktion **`void wait(uint32_t ms)`**, die **`ms`** Millisekunden wartet. Die Präprozessorkonstante **`LOOPS_PER_MS`** gibt an, wieviele Schleifendurchläufe gewartet werden muss, um eine Millisekunde verstreichen zu lassen.
- Meldet der Sensor im aktiven Betrieb einen leeren Wassertank, so wechselt die Kaffeemaschine in den Standby-Modus. Dazu werden Pumpe und Heizung ausgeschaltet und die Statusanzeige zeigt **`Standby`**. Alternativ kann die Kaffeemaschine auch jederzeit per Tastendruck ausgeschaltet werden.
- Sowohl Tastendrücke als auch Änderungen des Wasserstandes sollen mit Hilfe von Interrupts erkannt werden. (kein Polling!)
- Während des Wartens auf Tastendrücke oder Sensormeldungen soll der Mikrocontroller zum Stromsparen in den Schlafmodus gehen.



Information über die Hardware

Sie dürfen diese Seite zur besseren Übersicht bei der Programmierung heraustrennen!

Pumpe und Heizung: **PORTB**, Pin 2
- active-low: Pumpe und Heizung laufen sobald ein LOW-Pegel anliegt.
- Pin als Ausgang konfigurieren: entsprechendes Bit in **DDRB**-Register auf 1
- Pumpe und Heizung zu Beginn ausschalten; entsprechendes Bit in **PORTB**-Register auf 1

Taster: Interruptleitung **INT0** an **PORTD**, Pin 2
- active-low: Wird der Taster gedrückt, so liegt ein LOW-Pegel an.
- Pin als Eingang konfigurieren; entsprechendes Bit in **DDRD** Register auf 0
- Internen Pull-Up-Widerstand aktivieren; entsprechendes Bit in **PORTD**-Register auf 1
- Externe Interruptquelle **INT0**, ISR-Vektor-Makro: **INT0_vect**.
- Aktivierung der Interruptquelle erfolgt durch Setzen des **INT0**-Bits im Register **GICR**.

Sensor: Interruptleitung **INT1** an **PORTD**, Pin 3
- active-low: enthält der Tank Wasser, so liegt ein LOW-Pegel an.
- Pin als Eingang konfigurieren; entsprechendes Bit in **DDRD**-Register auf 0
- Internen Pull-Up-Widerstand aktivieren; entsprechendes Bit in **PORTD**-Register auf 1
- Externe Interruptquelle **INT1**, ISR-Vektor-Makro: **INT1_vect**.
- Aktivierung der Interruptquelle erfolgt durch Setzen des **INT1**-Bits im Register **GICR**.
- Abfrage des aktuellen Pegels durch entsprechendes Bit in **PIND**-Register
  (0 = nicht leer; 1 = leer)

Konfiguration der externen Interruptquellen 0 und 1 (Bits in Register **MCUCR**)

Interrupt 0		**Beschreibung**	**Interrupt 1**	
ISC01	**ISC00**		**ISC11**	**ISC10**
0	0	Interrupt bei low Pegel	0	0
0	1	Interrupt bei beliebiger Flanke	0	1
1	0	Interrupt bei fallender Flanke	1	0
1	1	Interrupt bei steigender Flanke	1	1

Ergänzen Sie das folgende Codegerüst so, dass ein vollständig übersetzbares Programm entsteht.

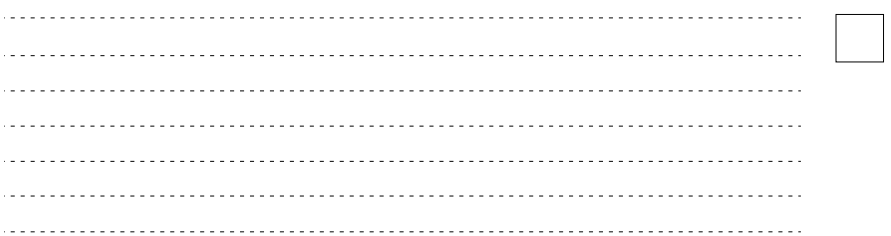

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <stdint.h>

#define LOOPS_PER_MS 100

typedef enum {
    Standby = 0,
    Active  = 1,
    NoWater = 2
} LEDState;

void setLEDState(LEDState state);


/* Funktionsdeklarationen, globale Variablen, etc. */
```



```
/* Unterbrechungsbehandlungsfunktionen */
```

A:

```
/* Funktion main */

   /* Variablen, Initialisierung */

  /* Hauptschleife */

      /*  Warten auf Ereignisse */
```

K:


```
      /*  Ereignis Tastendruck behandeln */
```

T:

```
/*  Ereignis Sensoränderung behandeln */
```

```
/* Ende main */

/* Funktion wait */
```

```
/* Ende wait */
```

SW:


```
/* Initialisierungsfunktion */
```

I:

Aufgabe 2b: Mail-Programm (15 Punkte)

Sie dürfen diese Seite zur besseren Übersicht bei der Programmierung heraustrennen!

Schreiben Sie ein Programm **`mail`**, das den Versand einer E-Mail an mehrere Empfänger übernimmt.

Als Parameter erhält Ihr Programm den Pfad zu einer E-Mail-Datei:

```
./mail invitation.txt
```

Diese Datei enthält in der ersten Zeile die (durch Komma getrennten) Empfänger und in den folgenden Zeilen den Inhalt der E-Mail:

```
To: homer@moes.com, marge@home.org, lisa@greenpeace.org, maggie@simpsons.com
Zeile 1 der E-Mail steht hier...
Zeile 2 der E-Mail steht hier...
Zeile 3 der E-Mail steht hier...
```

Ihr Programm soll nun die einzelnen Empfänger der E-Mail extrahieren und durch mehrfachen Aufruf von

```
/usr/bin/sendmail <ZIEL-ADRESSE> invitation.txt
```

diese E-Mail an jeden Empfänger zustellen.

Das Programm soll im Detail wie folgt funktionieren:

- Das Programm **`mail`** prüft zu Beginn, ob ein Parameter übergeben wurde. Sollte dies nicht der Fall sein, gibt es eine entsprechende Fehlermeldung aus und beendet sich.
- Wurde das Programm korrekt aufgerufen, öffnet es die angegebene Datei (zum Lesen) und liest die erste Zeile der Datei ein.
- Das Präfix "To: " am Anfang der Zeile wird übersprungen.
- Nun werden die einzelnen Empfänger der E-Mail nacheinander extrahiert und separat bearbeitet.
- Für das eigentliche Versenden wird ein Kindprozess erzeugt, der anschließend das Programm **`/usr/bin/sendmail`** mit der E-Mail-Adresse des Empfängers und der E-Mail-Datei als Argument aufruft.
- Der Vaterprozess wartet auf die Beendigung des Kindprozesses. Im Anschluss fährt er mit der Abarbeitung der Empfängerliste fort, bis die E-Mail an alle Empfänger versandt wurde.

Hinweise:

- Achten Sie auf eine korrekte Fehlerbehandlung der verwendeten Funktionen.
- Die Ausgabe von Fehlern soll auf dem stderr-Kanal erfolgen.
- Es kann davon ausgegangen werden, dass die erste Zeile der Datei immer vorhanden ist und nicht mehr als 1000 Zeichen enthält.

Ergänzen Sie das folgende Codegerüst so, dass ein vollständig übersetzbares Programm entsteht.


```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/wait.h>

// Funktion main

   // Argumente prufen

   // Datei oeffnen und erste Zeile lesen
```

```
  // "To: " überspringen
```

```
   // An alle Adressen versenden
```

```
// Ende main
```

M:

Aufgabe 3: (16 Punkte)

Die folgenden Beschreibungen sollen kurz und prägnant erfolgen (Stichworte, kurze Sätze)

a) Ein Mikrocontroller-Programm "main.c" verwende die Funktionen

```
void initTemp(uint8_t updateFreq);
int8_t readTemp(void);
```

eines Temperatursensor-Moduls. Das Modul implementiert diese Funktionen und verwaltet zudem einen Speicher für den zuletzt gelesenen Wert sowie eine Unterbrechungsbehandlungsfunktion, die diesen setzt:

```
int8_t lastValue;
void update_irq(void) {
     lastValue = ...;
}
```

Skizzieren Sie, welche Dateien (temp.*) erforderlich sind, um das Temperatursensor-Modul bereit zu stellen und im Hauptprogramm verwenden zu können. Was ist zu beachten (d. h. "was muss in welcher Datei wie stehen"), um eine korrekte Kapselung der Modulinterna sowie die korrekte Sichtbarkeit und Verwendung der Modulschnittstellen zu gewährleisten? (7 Punkte)

b) Skizzieren Sie die Schritte, die durchzuführen sind, um aus einem C-Quellprogramm ein lauffähiges Programm zu erstellen und es auf einem Mikrocontroller auszuführen. Nennen Sie die einzelnen Schritte und erklären Sie jeweils, was dabei passiert. (6 Punkte)

c) Beschreiben Sie das Lost-Update-Problem anhand eines konkreten Beispiels! (3 Punkte)

Aufgabe 4: (11 Punkte)

a) Beschreiben Sie die Unterschiede zwischen der Ausführung eines Programms auf einem Mikrocontroller ohne Betriebssystem und auf einem Betriebssystem wie z. B. Linux. (8 Punkte)

b) Nennen Sie drei Beispiele für Ressourcen, die ein Betriebssystem (wie z. B. Linux) im Prozesskontrollblock für jeden Prozess verwaltet. (3 Punkte)